Tabela de Escrutínio (Hashing)

Charles Ribeiro Chaves - 122086950
Filipe Viana da Silva - 121050053
Vinícius Brasil de Oliveira Barreto - 120029237

6 de julho de 2025

Sumário

1 Introdução

Esta atividade propõe a implementação de uma estrutura de dados baseada em tabela hash para eliminar duplicatas em datasets, utilizando programação orientada a objetos em Python. O objetivo é demonstrar a eficiência da tabela hash na deduplicação de registros, tarefa fundamental no pré-processamento de dados em Ciência de Dados.

2 Estruturas de Dados Utilizadas

- **Tabela Hash:** Implementada como uma lista de listas (encadeamento externo), permitindo acesso eficiente por índice e resolução de colisões.
- **Tupla:** Utilizada como chave de busca, composta pelos campos relevantes do dataset (ex: (`nome`, `ano`)).
- **Dicionário:** Cada registro do dataset é representado como um dicionário, facilitando o acesso aos campos por nome.

3 Divisão de Módulos e Organização

O código foi dividido em três partes principais:

1. **Classe `TabelaHash`:** Responsável por toda a lógica da tabela hash, incluindo inserção, busca, remoção e iteração sobre os elementos.
2. **Função `gerar_csv_aleatorio`:** Gera um arquivo CSV com registros aleatórios e duplicatas intencionais para testes.
3. **Função `eliminar_duplicatas_csv`:** Lê o arquivo CSV, insere os registros na tabela hash e retorna apenas os registros únicos.

4 Descrição das Rotinas e Funções

4.1 Classe `TabelaHash`

- **Construtor:** Permite definir o tamanho da tabela e uma função de hash customizada.
- `hash_padrao`: Função de hash baseada na função nativa do Python.
- `inserir`: Insere um registro se a chave ainda não existir (deduplicação).
- `buscar`: Retorna o dado associado à chave, se existir.
- `remover`: Remove um registro pela chave.
- `elementos`: Itera sobre todos os elementos únicos da tabela.

4.2 Função `gerar_csv_aleatorio`

Gera registros aleatórios com nomes e anos, adicionando duplicatas propositalmente. Escreve os registros em um arquivo CSV para facilitar o teste da deduplicação.

4.3 Função `eliminar_duplicatas_csv`

Lê o arquivo CSV, monta a chave de deduplicação a partir dos campos definidos, insere os registros na tabela hash e retorna a lista de registros únicos.

5 Complexidade de Tempo e Espaço

- **Tempo:** Inserção e busca na tabela hash são, em média, $O(1)$ por operação. A deduplicação de n registros ocorre em $O(n)$ no melhor caso.
- **Espaço:** O espaço utilizado é proporcional ao número de registros únicos, além do overhead das listas para encadeamento.

6 Problemas e Observações Durante o Desenvolvimento

- **Função de Hash:** A função de hash padrão do Python foi suficiente para o domínio do problema, mas pode ser substituída por outras funções conforme a natureza dos dados.
- **Colisões:** O encadeamento externo (listas) foi eficiente para resolver colisões, mantendo a simplicidade do código.
- **Flexibilidade:** O código permite fácil adaptação para diferentes campos de deduplicação e funções de hash.
- **Testes:** A geração automática de CSV facilitou a validação da solução.
- **Limitações:** Para datasets extremamente grandes, pode ser necessário ajustar o tamanho da tabela para evitar excesso de colisões.

7 Conclusão

A implementação da tabela hash orientada a objetos atendeu plenamente aos requisitos da atividade, proporcionando uma solução eficiente para a deduplicação de dados em arquivos CSV. A abordagem reduz a complexidade do processo para $O(n)$, tornando-a adequada para grandes volumes de dados. O código é modular, flexível e pode ser facilmente adaptado para diferentes domínios e funções de hash. A geração automática de dados e a deduplicação demonstraram, na prática, a eficácia da estrutura de dados hash para problemas reais de Ciência de Dados.